CHAPTER 1 • BASIC STRUCTURE OF COMPUTERS

PROBLEMS

1.1 List the steps needed to execute the machine instruction

Add LOCA,R0

in terms of transfers between the components shown in Figure 1.2 and some simple control commands. Assume that the instruction itself is stored in the memory at location INSTR and that this address is initially in register PC. The first two steps might be expressed as

- Transfer the contents of register PC to register MAR.
- Issue a Read command to the memory, and then wait until it has transferred the requested word into register MDR.

Remember to include the steps needed to update the contents of PC from INSTR to INSTR+1 so that the next instruction can be fetched.

1.2 Repeat Problem 1.1 for the machine instruction

Add R1,R2,R3

which was discussed in Section 1.6.3.

1.3 (*a*) Give a short sequence of machine instructions for the task: “Add the contents of memory location A to those of location B, and place the answer in location C.” Instructions

Load LOC,R*i*

and

Store R*i*,LOC

are the only instructions available to transfer data between the memory and generalpurpose register R*i* . Add instructions were described in Sections 1.3 and 1.6.3. Do not destroy the contents of either location A or B.

(*b*) Suppose that Move and Add instructions are available with the format

Move/Add Location1,Location2

These instructions move or add a copy of the operand at the first location to the second location, overwriting the original operand at the second location. Location*i* can be in either the memory or the processor register set. Is it possible to use fewer instructions to accomplish the task in Part *a*? If yes, give the sequence.

1.4 (*a*) Section 1.5 discusses how the input and output steps of a collection of programs such as the one shown in Figure 1.4 could be overlapped to reduce the total time needed to execute them. Let each of the six OS routine execution intervals be 1 unit of time, with each disk operation requiring 3 units, printing requiring 3 units, and each program execution interval requiring 2 units of time. Compute the ratio of best overlapped time to nonoverlapped time for a long sequence of programs. Ignore start-up and ending transients.

(*b*) Section 1.5 indicated that program computation can be overlapped with either input or output operations or both. Ignoring the relatively short time needed for OS routines, what is the ratio of best overlapped time to nonoverlapped time for completing the execution of a collection of programs, where each program has about equal balance among input, compute, and output activities?

1.5 (*a*) Program execution time, *T*, as defined in Section 1.6.2, is to be examined for a certain high-level language program. The program can be run on a RISC or a CISC computer. Both computers use pipelined instruction execution, but pipelining in the RISC machine is more effective than in the CISC machine. Specifically, the effective value of *S* in the *T* expression for the RISC machine is 1.2, but it is only 1.5 for the CISC machine. Both machines have the same clock rate, *R*. What is the largest allowable value for *N*, the number of instructions executed on the CISC machine, expressed as a percentage of the *N* value for the RISC machine, if time for execution on the CISC machine is to be no longer than that on the RISC machine?
(*b*) Repeat Part *a* if the clock rate, *R*, for the RISC machine is 15 percent higher than that for the CISC machine.

1.6 (*a*) A processor cache, as shown in Figure 1.5, is discussed in Section 1.6. Suppose that execution time for a program is directly proportional to instruction access time and that access to an instruction in the cache is 20 times faster than access to an instruction in the main memory. Assume that a requested instruction is found in the cache with probability 0.96, and also assume that if an instruction is not found in the cache, it must first be fetched from the main memory to the cache and then fetched from the cache to be executed. Compute the ratio of program execution time without the cache to program execution time with the cache. This ratio is usually defined as the speedup factor resulting from the presence of the cache.

(*b*) If the size of the cache is doubled, assume that the probability of not finding a requested instruction there is cut in half. Repeat Part *a* for a doubled cache size.